What is a Provenance Pattern?

Dave Dubin

March 22, 2018

What is a “provenance pattern?”

- A provenance pattern is an abstract generalization of a solution for a use case.

What is a "provenance pattern?"

- A provenance pattern is an abstract generalization of a solution for a use case.
- More generally, a solution pattern represents an interpretive response to a use case, highlighting key decisions and recommendations.

What is a "provenance pattern?"

- A provenance pattern is an abstract generalization of a solution for a use case.
- More generally, a solution pattern represents an interpretive response to a use case, highlighting key decisions and recommendations.
- Use case example from the TDWG WG: How should we represent agent roles in provenance documentation?

What is a "provenance pattern?"

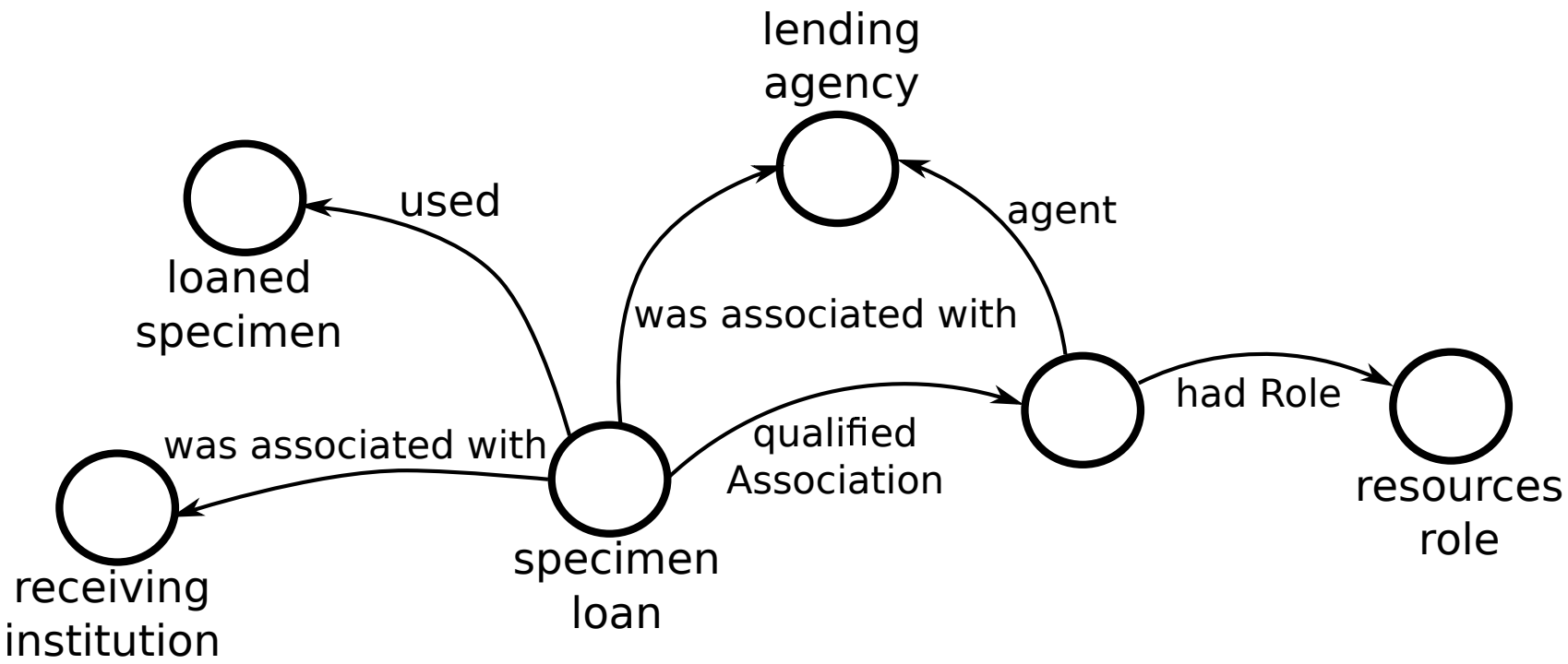
- A provenance pattern is an abstract generalization of a solution for a use case.
- More generally, a solution pattern represents an interpretive response to a use case, highlighting key decisions and recommendations.
- Use case example from the TDWG WG: How should we represent agent roles in provenance documentation?
- Scenario: an agency lends a physical specimen to a researcher. How shall we document the agency's role in contributing to the research by providing the loan?

What is a "provenance pattern?"

- A provenance pattern is an abstract generalization of a solution for a use case.
- More generally, a solution pattern represents an interpretive response to a use case, highlighting key decisions and recommendations.
- Use case example from the TDWG WG: How should we represent agent roles in provenance documentation?
- Scenario: an agency lends a physical specimen to a researcher. How shall we document the agency's role in contributing to the research by providing the loan?
- We have IDs for the loaned specimen, the lending institution, and the receiving institution. We propose to reify the loan itself as a transaction, and role taxonomies like CRediT[1] provide standardized labels (like "resources role").

[1] http://docs.casrai.org/CRediT

Documenting agent roles using PROV



The PROV illustration

Things to notice about the PROV solution:

1. The CRediT role is an attribute of the *association* between the lending agency and the loan event; that's how the PROV data model works.

The PROV illustration

Things to notice about the PROV solution:

1. The CRediT role is an attribute of the *association* between the lending agency and the loan event; that's how the PROV data model works.
2. But since RDF admits only binary relationships, we reify the association as a vertex as well as an edge.

The PROV illustration

Things to notice about the PROV solution:

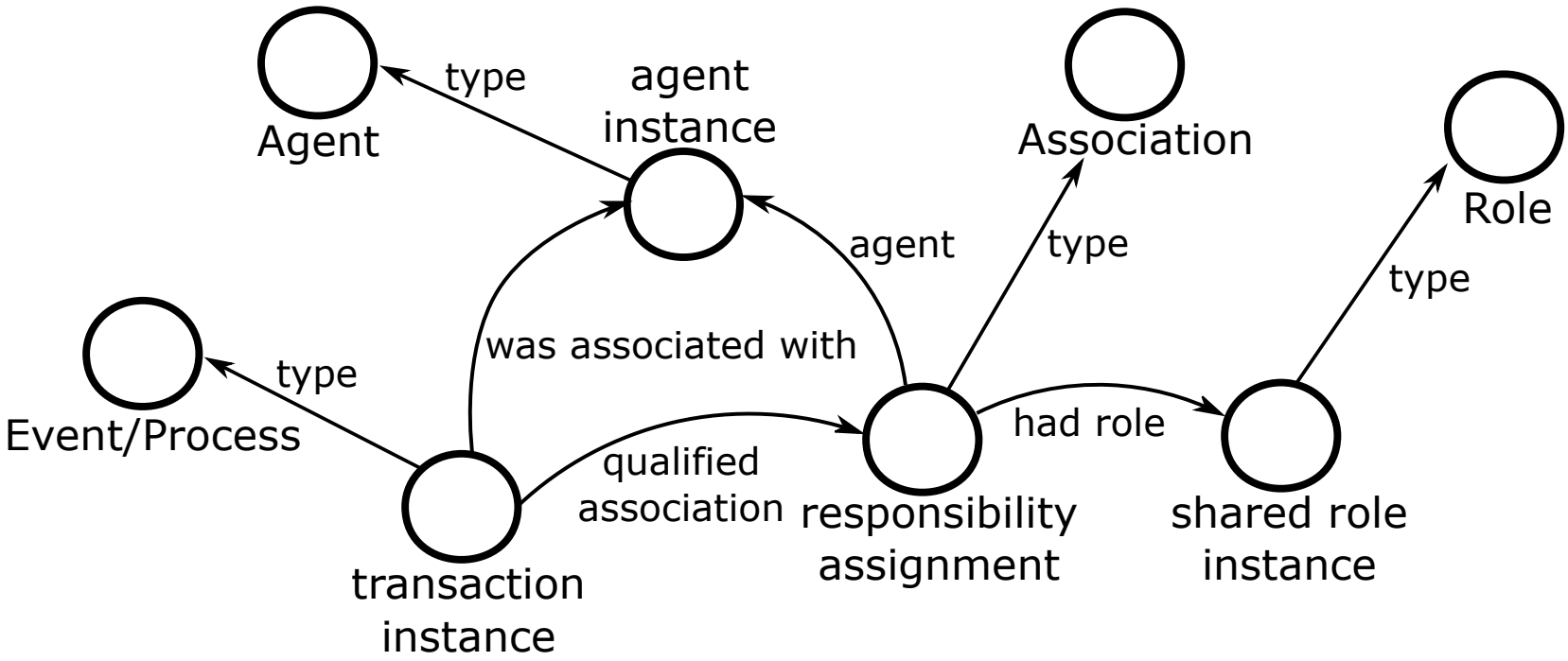
1. The CRediT role is an attribute of the *association* between the lending agency and the loan event; that's how the PROV data model works.
2. But since RDF admits only binary relationships, we reify the association as a vertex as well as an edge.
3. The **resources role** in this example is a *particular,* i.e., an instance of a **Role** class.

The PROV illustration

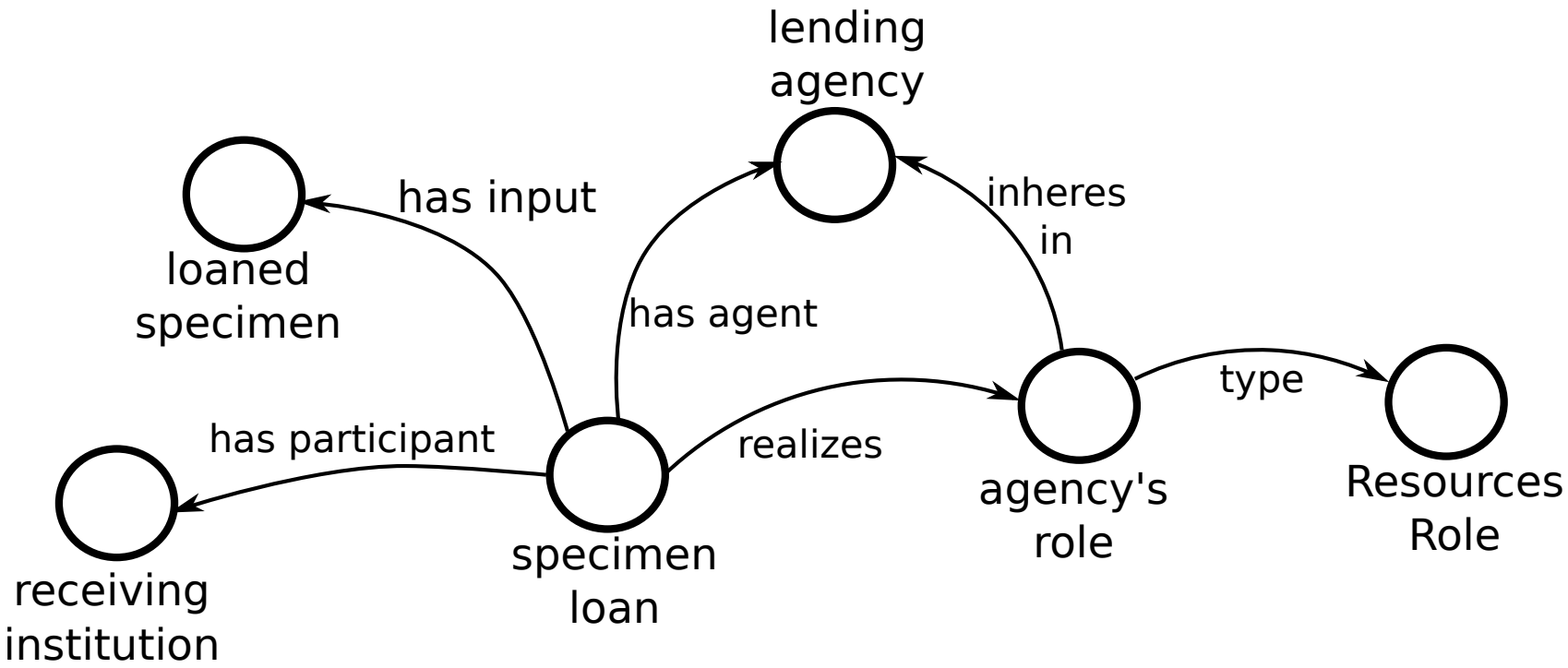
Things to notice about the PROV solution:

1. The CRediT role is an attribute of the *association* between the lending agency and the loan event; that's how the PROV data model works.
2. But since RDF admits only binary relationships, we reify the association as a vertex as well as an edge.
3. The **resources role** in this example is a *particular,* i.e., an instance of a **Role** class.
4. Different lending agencies could play that very same **resources role** in the context of other loan events.

Pattern 1: roles as shared social object instances



The VIVO-ISF illustration



The VIVO-ISF illustration

Things to notice about the VIVO-ISF solution:

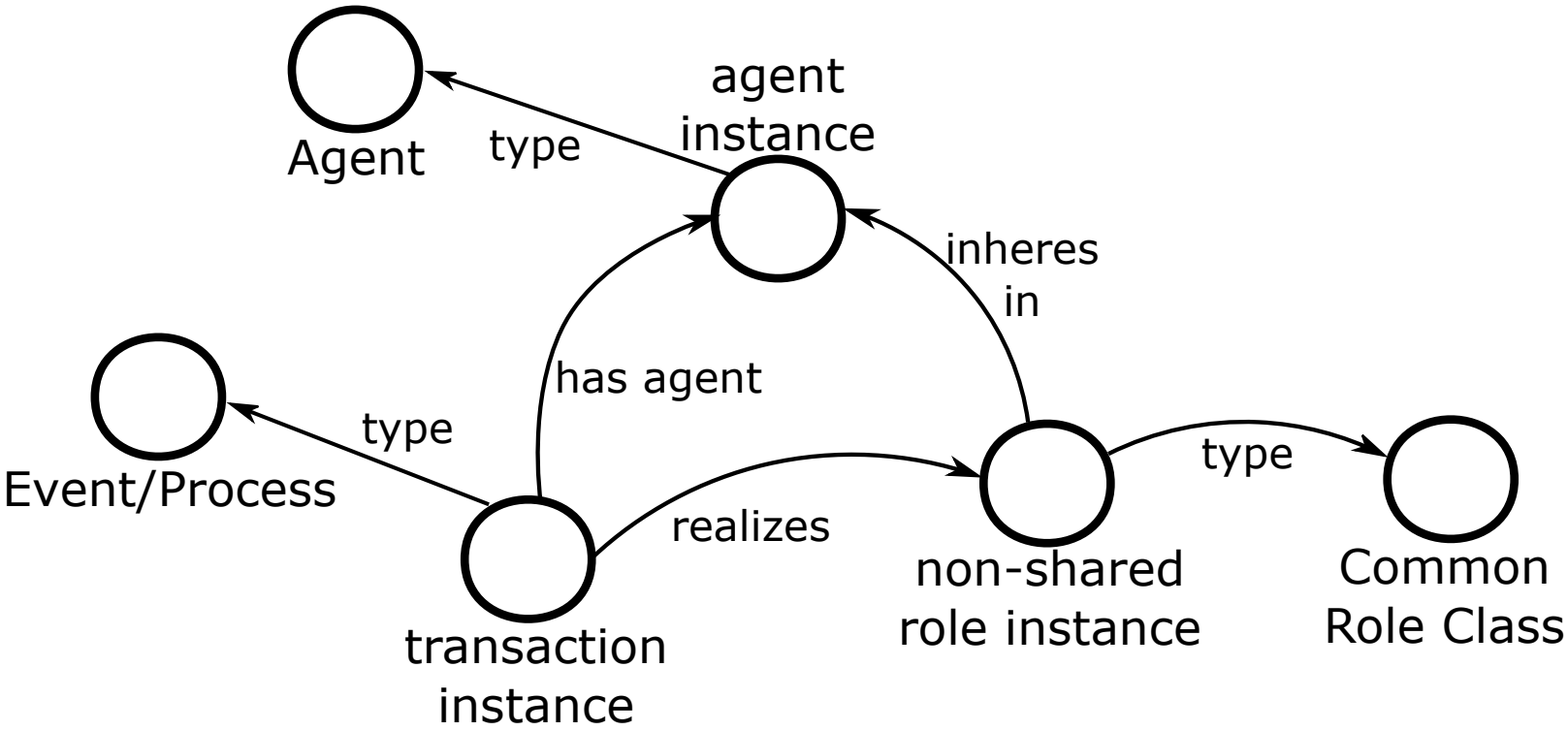
1. VIVO-ISF is based in BFO, the Basic Formal Ontology, which has an Aristotelian characterization of roles. Roles inhere within the agents that assume them.

The VIVO-ISF illustration

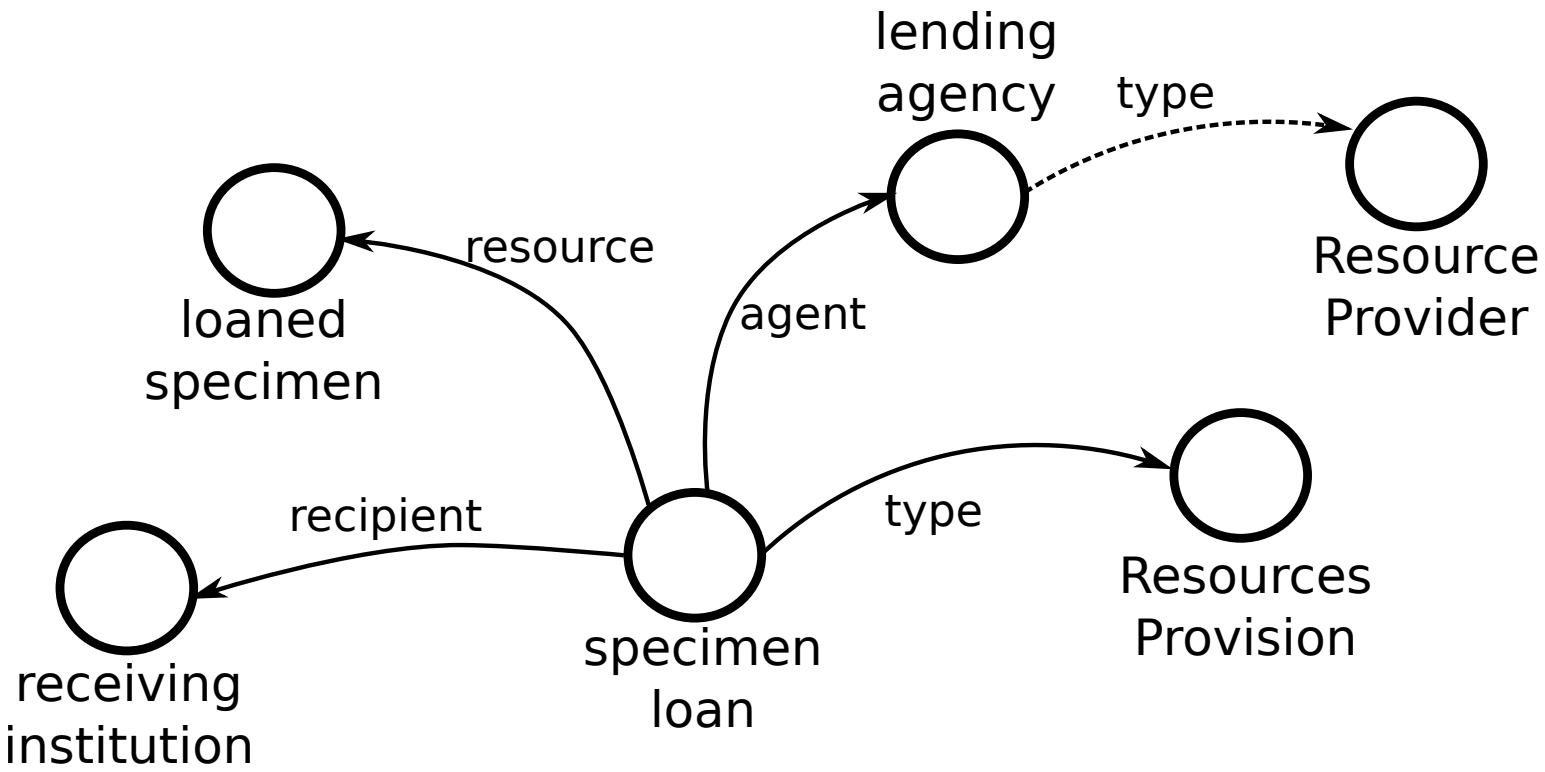
Things to notice about the VIVO-ISF solution:

1. VIVO-ISF is based in BFO, the Basic Formal Ontology, which has an Aristotelian characterization of roles. Roles inhere within the agents that assume them.
2. The **Resources Role** in this example is not an instance or particular, but a *class.* That is because no role instance can inhere within two different agents. Two agent roles may have the same *class identity,* but not the same *individual identity.*

Pattern 2: roles as a common class identity



The contingent subclass pattern



A third pattern: roles as contingent agent subclasses

- The last example illustrates a third way we can represent contributor roles.

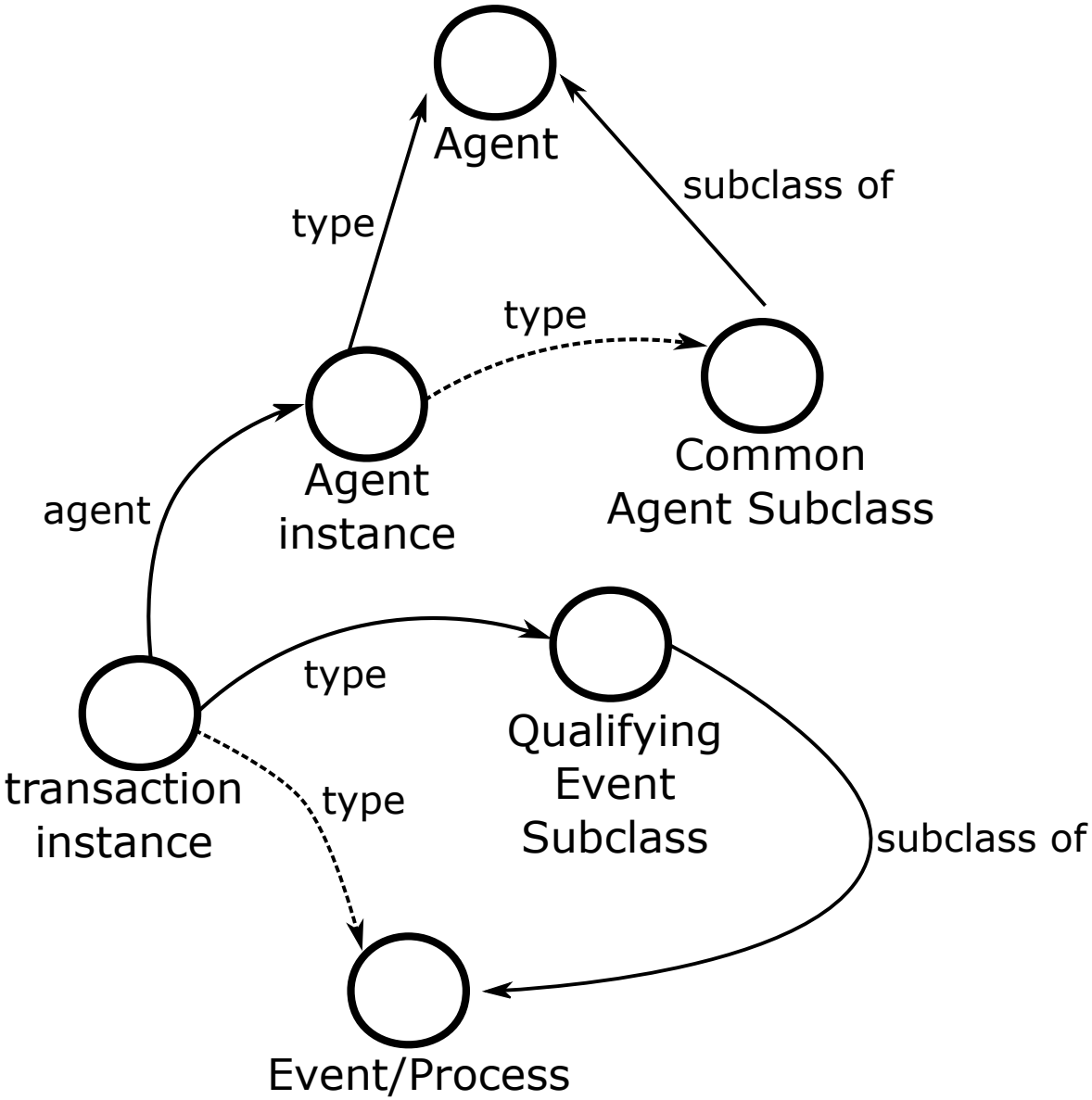
A third pattern: roles as contingent agent subclasses

- The last example illustrates a third way we can represent contributor roles.
- Being a **Resource Provider** is understood as a class identity for the agents themselves. This class identity is deduced (dashed edge) from the agent's participation in a **Resources Provision** event.

A third pattern: roles as contingent agent subclasses

- The last example illustrates a third way we can represent contributor roles.
- Being a **Resource Provider** is understood as a class identity for the agents themselves. This class identity is deduced (dashed edge) from the agent's participation in a **Resources Provision** event.
- The participation that licenses this inference is not directly associated with the triggering event (as it was in the realization of the BFO role).

Pattern 3: contingent subclasses



Justifying the choice of a pattern

- In the running example, pattern choice is justified on the basis of stakeholder communities with whom one wishes to cooperate.

Justifying the choice of a pattern

- In the running example, pattern choice is justified on the basis of stakeholder communities with whom one wishes to cooperate.
- Computational limitations or costs can also factor into a decision to adopt or reject a pattern.

Justifying the choice of a pattern

- In the running example, pattern choice is justified on the basis of stakeholder communities with whom one wishes to cooperate.
- Computational limitations or costs can also factor into a decision to adopt or reject a pattern.
- It's not always a matter of chosing one pattern over another: in the running example the bare facts are the same, only the interpretations differ.

Summary

1. Solution patterns are our WG's strategic response to problems of communication and adoption for our outputs.

Summary

1. Solution patterns are our WG's strategic response to problems of communication and adoption for our outputs.
2. Since use cases are vivid for many RDA working and interest groups, we try to make our recommendations match the granularity of a use case.

Summary

1. Solution patterns are our WG's strategic response to problems of communication and adoption for our outputs.
2. Since use cases are vivid for many RDA working and interest groups, we try to make our recommendations match the granularity of a use case.
3. Our solution patterns typically don't match the abstraction level of the the use case, but we aim for a level that highlights key modeling and encoding decisions.

Summary

1. Solution patterns are our WG's strategic response to problems of communication and adoption for our outputs.
2. Since use cases are vivid for many RDA working and interest groups, we try to make our recommendations match the granularity of a use case.
3. Our solution patterns typically don't match the abstraction level of the the use case, but we aim for a level that highlights key modeling and encoding decisions.
4. Patterns are linked in our database to associated use cases (and/or to other patterns at other levels of generality). For example, all three of the agent role patterns have direct counterparts for object or event roles (e.g., "instrument" or "replication," respectively).